In 2000 we looked at a number of tools and techniques for making Linux systems and networks more secure. It occurs to me that I've made one assumption that's horrendously incorrect − that my readers have already gone through the basic steps in securing their systems.

I want to take at least one month to go over some basic checks that should be performed on all of your Linux systems. If you haven't addressed the following security checks, using ssh, a firewall, or any other security techniques won't help.

Passwords

This is the most important one of all. It's very simple and *very* important. If you use badly chosen passwords (or worse yet, fail to assign passwords any at all), you have a system that essentially **can't** be secured.

What's the point of putting steel bars on the windows if your front door has no locks? :−)

Here are Bill's guidelines for choosing passwords.

- Passwords should be at least 7 characters long.
- They should have at least three of the following: uppercase letters, lowercase letters, digits, and other characters.
- They should **not** be: names (especially family or pet names), words, the same as the account name, or any of the preceding spelled backwards. Don't use "password" or "god" − these are equivalent to sending out invitations to hackers to break into your system!
- Don't let anyone watch while you type in a password. I've gotten in a habit of specifically looking away whenever someone's typing in their password, and ask others to extend me the same courtesy. It makes no difference how much I trust the other individual or how much I think I'm worthy of their trust − passwords are used to identify an individual.
- Never give out your password. **Never.**
- Never write your passwords down and leave them near your machine.
- Passwords should be changed no more frequently than once a month and no less frequently than once every 6 months.
- The above rules are especially important for the root or administrator accounts on a given system. Make them longer, more random, and less like words.

Passphrases

If you use pgp or ssh (see my previous articles on ssh), you may already be using a passphrase to protect your pgp or ssh keys. In ssh, a single key and passphrase combination can provide access to lots of other accounts. PGP signed or encrypted documents are as close as we get to definitively identifying the author and/or restricting who can read a document.

For these reasons, all of the above rules apply to passphrases even more than to regular and root passwords. Passphrases should be as long as a sentence, but, again, not regular English. Use lots of mangled spelling, wierd punctuation, nonsense words, and mixed case.

"Passwords are like underwear. You don't share them, you don't hang them on your monitor, or under your keyboard, you don't email them, or put them on a web site, and you must change them very often."

Many thanks to Kurt Bemis for that quote from his .sig.

Package updates

I recently spent some time researching the Ramen worm (if you want to see if you've been attacked or need help cleaning up, see Ramenfind). This worm attacks only Redhat 6.2 and 7.0 machines, gaining access through the known vulnerabilities in wu−ftpd, rpc.statd, and LPRng.

There's a very important word there: **known**. RedHat released fixes for these holes on 6/23/2000, 6/17/2000, and 10/4/2000, respectively. Ramen was released in mid January; 3 months after the most recent fix. If everyone administering a Redhat 6.2 or 7.0 system had applied those updates, Ramen could not have affected a single system.

While distribution maintainers do their best to release secure software, new vulnerabilities **will** show up after the CD is shipped. You need to apply the fixes to your system when they show up.

Here's how − it's easy.

- Sign up for your distribution's security announcements list. Take a look on the web site for "mailing lists", "security", "errata", or "updates". The number of messages to these lists is usually less than 10 a month.
- Locate a mirror site containing the updates for your distribution. For example, I use Redhat 7.0 and usually go to ftp://ftp.varesearch.com/pub/mirrors/redhat/redhat/updates/7.0/ . Inside that directory are three directories relevant to me; noarch (the "no specific architecture" files like text files or perl scripts), the "i686" directory (because I have a Pentium III), and the i386 directory that holds rpms that will work with any Intel chip. If you're not sure what kind of processor you have, skip the contents of the i586 and i686 directories and just get files from i386; they'll still work, but just a hair slower.
- Pull down any updates for packages you have installed from these ftp directories. For example, there's a file called "fetchmail−5.5.0−3.i386.rpm" in the i386 directory. I check my system with the "rpm −q fetchmail" command and I get back "fetchmail−5.5.0−2", showing that I have an earlier version of that program, so I need to get the update. If you don't feel like checking which ones you have, simply pull down the entire tree to a local directory. The next time you go to check on updates you can just pull down the ones you don't have.
- Go to the directory with your updates and run the following command:

  ```
  rpm −F −vh *.rpm
  ```

  This instructs the package manager to "**F**reshen" the packages in this directory (upgrade them, but only if you already have an older version of the package in your system), do it **v**erbosely, and print **h**ash marks showing its progress.

  If "rpm −F −vh" replies with the following error:

  ```
  −F: unknown option
  ```

  , your version of rpm doesn't know how to only install the update if the old version is already installed. Before installing each update, you'll have to check by hand if that package is installed.
- Occasionally the default configuration files in these packages will change between releases. When this happens, rpm will report that it has moved your version out of the way with a message such as

  ```
  [root@sparrow updates−7.0]# rpm −Uvh php−4.0.4pl1−3.i386.rpm
  warning: /etc/php.ini saved as /etc/php.ini.rpmsave
  ```

```
        php                         ##################################################
        [root@sparrow updates−7.0]#
```

This is a warning that you should carry the changes you've made to /etc/php.ini from your /etc/php.ini.rpmsave to the new /etc/php.ini .

The above steps really don't take very long − when you're administering a single system. What happens when you're administering 20 or 30 systems with a mix of distributions?

Redhat provides a way for the updates to happen automatically. The "up2date" package will do these upgrades for you. You'll need to sign up for Redhat's priority service − but this may well be worth it if you can't find the time to do the upgrades by hand.

Other distributions may provide similar automatic upgrade procedures; check with them. If nothing else, you can sign up with individuals or companies that will do your upgrades for you.

Open Ports

Here's the last big problem in default system setup − open ports.

When a server program, like the Apache web server or the Bind DNS server, wishes to accept connections from other machines, they listen on one or more ports (TCP port 80 for Apache and UDP port 53 and TCP port 53 for Bind). When packets arrive at this machine destined for one of these listening ports, the kernel and libraries direct these incoming requests up to the program listening there. That program can then start a conversation with the machine that sent the request.

Let's say I install 10 different server applications when I do my base Linux install. Those servers listen on 12 different ports. So far, there's no problem with this. But say that a month from now, someone on the Internet discovers that there's a security hole in the Gooberd server. By connecting to that server and typing 1025 spaces in response to the "Username:" prompt, it's possible to get shell access to the box without a password. If Gooberd was one of those servers I installed, my box can be attacked as soon as the exploit is made public. Worse yet, some attackers scan for open ports on your system far in advance of exploits being made available and keep a database of machines with port 1 open, machines with port 2 open, etc. When the exploit comes out, they have a ready made list of systems with that port open so they can attack thousands of machines instantly.

sigh

The logical approach is to close as many of these ports as possible. If I can cut the list of 12 open ports on my system − most of which didn't have to be open in the first place − to only 2 ports for servers I really do want running, I've cut my chances of being successfully attacked by a future exploit by a factor of 6. I'm still vulnerable, but much less vulnerable. I still have to keep a close eye on the announcement lists for those two servers, but now don't need to worry about the other ten.

First, let's see what ports are open on your system. The "netstat" program will list them for us. Run the following as root:

```
bash# netstat −a −p
Active Internet connections (servers and established)
Proto Recv−Q Send−Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 *:login                 *:*                     LISTEN      464/inetd
tcp        0      0 *:linuxconf             *:*                     LISTEN      464/inetd
tcp        0      0 *:shell                 *:*                     LISTEN      464/inetd
tcp        0      0 *:finger                *:*                     LISTEN      464/inetd
tcp        0      0 *:sunrpc                *:*                     LISTEN      348/portmap
tcp        0      0 *:www                   *:*                     LISTEN      563/httpd
tcp        0      0 *:auth                  *:*                     LISTEN      428/identd
tcp        0      0 my.hostname:domain      *:*                     LISTEN      496/named
tcp        0      0 localhost:domain        *:*                     LISTEN      496/named
tcp        0      0 *:ftp                   *:*                     LISTEN      464/inetd
tcp        0      0 *:ssh                   *:*                     LISTEN      505/sshd
tcp        0      0 *:telnet                *:*                     LISTEN      464/inetd
tcp        0      0 *:squid                 *:*                     LISTEN      592/(squid)
tcp        0      0 *:smtp                  *:*                     LISTEN      538/sendmail: acc
tcp        0      0 dial-10-203-apx-01:1022 shell1.myisp.n:ssh      ESTABLISHED 4220/ssh
udp        0    220 *:1024                  *:*                                 496/named
udp        0      0 *:talk                  *:*                                 464/inetd
udp        0      0 *:ntalk                 *:*                                 464/inetd
udp        0      0 *:snmp                  *:*                                 480/snmpd
udp        0      0 my.hostname:domain      *:*                                 496/named
udp        0      0 localhost:domain        *:*                                 496/named
udp        0      0 localhost:32789         localhost:domain        ESTABLISHED 861/netscape-comm
udp        0      0 *:3130                  *:*                                 592/(squid)
udp        0      0 *:3401                  *:*                                 592/(squid)
udp        0      0 *:sunrpc                *:*                                 348/portmap
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node PID/Program name    Path
unix  2      [ ACC ]     STREAM     LISTENING     1840   496/named           /var/run/ndc
unix  12     [ ]         DGRAM                    832    401/syslogd         /dev/log
unix  2      [ ACC ]     STREAM     LISTENING     2519   620/xfs             /tmp/.font-unix/fs-1
unix  2      [ ]         DGRAM                    2633   640/login -- root
unix  2      [ ]         DGRAM                    2613   637/login -- root
unix  2      [ ]         DGRAM                    2571   638/login -- root
unix  2      [ ]         DGRAM                    2522   620/xfs
unix  2      [ ]         DGRAM                    2264   591/squid
unix  2      [ ]         DGRAM                    2047   538/sendmail: accep
unix  2      [ ]         DGRAM                    1861   505/sshd
unix  2      [ ]         DGRAM                    1838   496/named
unix  2      [ ]         DGRAM                    1173   428/identd
unix  2      [ ]         DGRAM                    998    412/klogd
unix  2      [ ]         STREAM     CONNECTED     347    1/init
```

If your version of netstat complains about the "−p" command line parameter, rerun it without the "−p". You'll lose the "PID/Program name" field, but you'll still see what ports are open.

First, you can ignore everything under "Active UNIX domain sockets" for this discussion as these are only visible to programs on the system and not to external machines. Secondly, the listening ports are the tcp and udp lines with "*.*" in the "Foreign Address" field. Here's that output again with only the interesting lines:

```
bash# netstat -a -p
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 *:login                 *:*                     LISTEN      464/inetd
tcp        0      0 *:linuxconf             *:*                     LISTEN      464/inetd
tcp        0      0 *:shell                 *:*                     LISTEN      464/inetd
tcp        0      0 *:finger                *:*                     LISTEN      464/inetd
tcp        0      0 *:sunrpc                *:*                     LISTEN      348/portmap
tcp        0      0 *:www                   *:*                     LISTEN      563/httpd
tcp        0      0 *:auth                  *:*                     LISTEN      428/identd
tcp        0      0 my.hostname:domain      *:*                     LISTEN      496/named
tcp        0      0 localhost:domain        *:*                     LISTEN      496/named
tcp        0      0 *:ftp                   *:*                     LISTEN      464/inetd
tcp        0      0 *:ssh                   *:*                     LISTEN      505/sshd
tcp        0      0 *:telnet                *:*                     LISTEN      464/inetd
tcp        0      0 *:squid                 *:*                     LISTEN      592/(squid)
tcp        0      0 *:smtp                  *:*                     LISTEN      538/sendmail: acc
udp        0    220 *:1024                  *:*                                 496/named
udp        0      0 *:talk                  *:*                                 464/inetd
udp        0      0 *:ntalk                 *:*                                 464/inetd
udp        0      0 *:snmp                  *:*                                 480/snmpd
udp        0      0 my.hostname:domain      *:*                                 496/named
udp        0      0 localhost:domain        *:*                                 496/named
udp        0      0 *:3130                  *:*                                 592/(squid)
udp        0      0 *:3401                  *:*                                 592/(squid)
udp        0      0 *:sunrpc                *:*                                 348/portmap
```

The port information can be found in columns 4 and 1. For example, the third line of output shows that something is listening on the TCP "login" port. If you're interested in what port number that is, see /etc/services:

```
bash# cat /etc/services | grep ^login
login           513/tcp
```

Running netstat with the "−n" parameter will also output numeric port information in addition to leaving IP addresses in numeric format. By the way − the reason that 1024/UDP, 3130/UDP and 3401/UDP are listed as numbers is that /etc/services doesn't have a name for those ports. Respectively, those are Bind's client port, Squid's inter−cache communications port (ICP), and Squid's SNMP port.

The last column shows what program is listening on that port. In this case, the "inetd" server is listening for incoming connections on port 513/TCP. As I don't want to make this service available across the network or Internet, I'd like to shut this off.

Here's how to close the ports on a Linux system. Either of the following approaches will work.

1. This approach completly removes the service.

   Since /etc/inetd.conf hints that the "login" port is serviced by the in.rlogind program, I can remove the rpm that holds that server:

   ```
   [root@sparrow /etc]# which in.rlogind
   /usr/sbin/in.rlogind
   [root@sparrow /etc]# rpm −qf /usr/sbin/in.rlogind
   rsh−server−0.17−2.2
   [root@sparrow /etc]# rpm −−erase rsh−server
   [root@sparrow /etc]# rpm −q rsh−server
   package rsh−server is not installed
   [root@sparrow /etc]# ls −al /usr/sbin/in.rlogind
   ls: /usr/sbin/in.rlogind: No such file or directory
   ```

   First, I find out what directory holds the server file. Then I find out which RPM installed this server and remove it. The last 2 queries just double check that the rpm and file are truly gone.

   If you use this approach, it's still a good idea to comment out the port line in /etc/inetd.conf (if appropriate; read on), or kill the server program with:

   ```
   killall −9 the_server_name
   ```
2. This approach disables the service, but leaves the daemon installed in case you need it later. Do one of

the following:

1. If the daemon is run out of inetd, add a "#" at the beginning of the line that starts it in /etc/inetd.conf . The original

   ```
   login   stream  tcp     nowait  root    /usr/sbin/tcpd  in.rlogind
   ```

   would become

   ```
   # login   stream  tcp     nowait  root    /usr/sbin/tcpd  in.rlogind
   ```
2. If it is started out of xinetd, change the "disable = no" line in /etc/xinetd.d/rlogin to "disable = yes".
3. If the service, say sendmail, is started from a startup script in /etc/rc.d/init.d or /etc/init.d , you can disable it for future boots by typing "chkconfig sendmail off", using the menu driven "ntsysv" program, or by renaming the following files:

   ```
   mv /etc/rc.d/rc3.d/S80sendmail /etc/rc.d/rc3.d/K30sendmail
   mv /etc/rc.d/rc5.d/S80sendmail /etc/rc.d/rc5.d/K30sendmail
   /etc/rc.d/init.d/sendmail stop
   ```

   The numbers 80 and 30 refer to the order in which services get started and shut down. If you're not sure what to use, subtract the existing number from 100.

   The last line shuts down the service gracefully now; remember that all of the above steps shut down the service on the *next* and future boots.

What services should be shut down on a Linux system? In short, unless you know you need a particular server, **all** of them in your netstat listing (don't shut off everything in /etc/rc.d/init.d, just the services listening on ports). I'm quite serious − a Linux system does not have to listen on any ports to function. Now obviously, if the system is serving as a mail server, you'll need to leave the sendmail service enabled and leave ipop3d and imapd servers enabled in inetd/xinetd. If it's serving up web pages, httpd will need to be left running, etc. If you're not sure, shut it off and see if any problems arise.

Once you've made all the changes, restart inetd or xinetd, whichever is running on your system:

```
/etc/rc.d/init.d/inetd restart
```

or

```
/etc/rc.d/init.d/xinetd restart
```

Closing notes

The difficulty in writing a short article on a topic as simple as shutting down open ports is the complexity of handling the differences in distributions; even just Redhat 6.2 and 7.0 differ enough to make it confusing! I apologize for the Redhat/Intel−centric view above, but hope you understand that I'm presenting the technique and hoping you can adapt the approach to your system.

There's a marvelous tool that presents a consistent interface to basic administration tasks on a wide range of distributions − Linuxconf . It has command line, gui, and web interfaces that transparently handles these inter−distribution differences. It's well worth a look if you haven't already checked it out. I have a lot of respect for Jacques Gelinas, Linuxconf's author, and his team.

William is an Open−Source developer, enthusiast, writer, and advocate from New Hampshire, USA. His day job at SANS pays him to work on network security and Linux projects.

Last updated 12/18/2003.